Android of Theseus: ~~Patching applications to improve analysis~~

Dynamic Analysis is Hard

Jean-Marie Mineau

2025-06-06

Outline

Context

Android

- Java-ish
- Can load classes dynamically
- Can do reflection
- Malware do that

Class Loading

```
InMemoryDexClassLoader classloader = new InMemoryDexClassLoader(dexbuffer, null);

Class clz = cl.loadClass("com.example.MyClass");
Method mth = clz.getMethod("myMethod", String.class, String.class);
String result = (String)mth.invoke(null, new Object[] {"method", "arg"});
```

Class Loading

```
InMemoryDexClassLoader classloader = new InMemoryDexClassLoader(dexbuffer, null);

Class clz = cl.loadClass("com.example.MyClass");
Method mth = clz.getMethod("myMethod", String.class, String.class);
String result = (String)mth.invoke(null, new Object[] {"method", "arg"});
```

```
String result = MyClass.myMethod("method", "arg");
```

The Project

Objectif

I want to modify an application that do dynamique loading so that:

Objectif

I want to modify an application that do dynamique loading so that:

- Code loaded dynamically is added to the application

Objectif

I want to modify an application that do dynamique loading so that:

- Code loaded dynamically is added to the application
- Method called with reflection are called directly

Objectif

I want to modify an application that do dynamique loading so that:

- Code loaded dynamically is added to the application
- Method called with reflection are called directly
- The application behavior remains unchanged

Step 1: Collect Runtime Data

I need the bytecode loaded dynamically.

I need the list of method called, and where they are called.

Step 1: Collect Runtime Data

FRIDA

- Instrumentation framework
- Inject a JavaScript intepretor in the memory of a process
- Can replace method calls by js scripts

Step 1.1: Collect Bytecode

Android always call either `DexFile.openInMemoryDexFilesNative()` or `DexFile.openDexFileNative()` to parse bytecode.

We can intercept this calls to get all the bytecode used by the application.

Step 1.2: Collect Reflection Data

We can intercept `Method.invoke()` and `Constructor.newInstance()` to get the name of the method and its argument. Easy.

Step 1.2: Collect Reflection Data

We can intercept `Method.invoke()` and `Constructor.newInstance()` to get the name of the method and its argument. ~~Easy~~.

Where is the method called?

Step 1.2: Collect Reflection Data

Historically, this information is collected using the error stack trace. Unfortunatly the exact location inside the method bytecode of the call is still unreliable.

Android SDK 34 introduced the `StackWalker` API that allow to travel the stack and collect the bytecode index of each call.

Step 2: The Application Transformation

What now?

Step 2: The Application Transformation

The bytecode can be added as additionnal `classes<n>.dex` files to the application

Step 2: The Application Transformation

One `invoke()` location can call different methods

Runtime data may be incomplete

Pathological Case

```
Object myInvoke(Method mth, Object oj, Object[] args) {
  return mth.invoke(obj, args);
}
```

Step 2: The Application Transformation

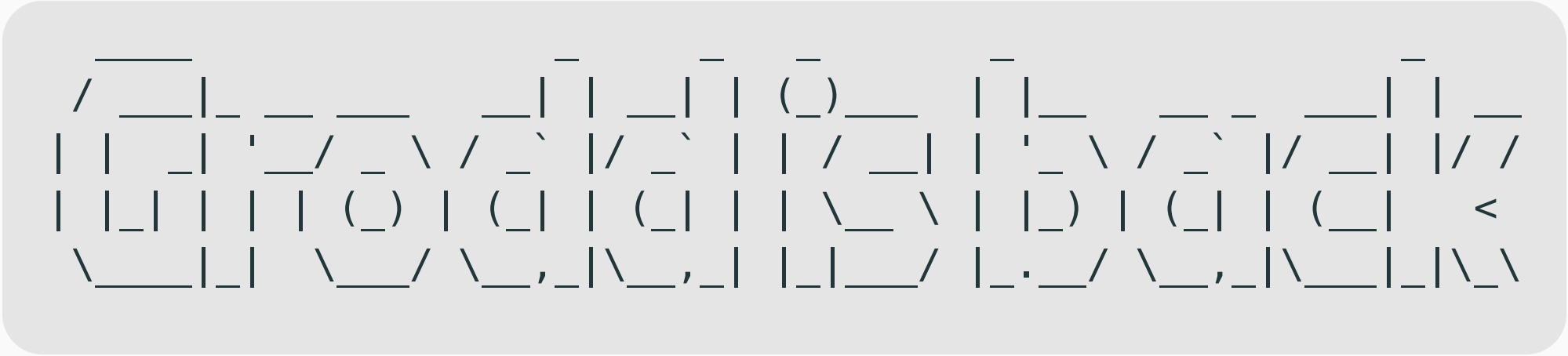
```
Object result = meth.invoke(obj, args);
```

```
Object result;
if T.check_is_Class1_method1(meth) {
  result = (Object)obj.method1((String)args[0])
} else if T.check_is_Class2_method2(meth) {
  result = (Object)obj.method2(
    ((Boolean)args[0]).booleanValue(),
    (String)args[1]
  );
} else {
  retult = meth.invoke(obj, args);
}
```

Step 3: Automation

We need something to explore the application automatically

Step 3: Automation



Step 3: Automation

Grodd Runner was extracted from GroddDroid, a previous project of the CIDRE team.

It uses `uiautomator` to explore the application like graph.

A Series of Unfortunate Events

Frida

Frida did not know how to parse the type `[[B` (Array of Array of byte)

`DexFile.openInMemoryDexFilesNative()` has `[[B` in its signature

Frida

Frida did not know how to parse the type `[[B` (Array of Array of byte)

`DexFile.openInMemoryDexFilesNative()` has `[[B` in its signature

PR merged to the Frida project, problem solved!

Dynamic Analysis

Dynamic Analysis

We took the naive approach:

- Google android emumator
- Rooted to run Frida
- Headless to run on a server

Dynamic Analysis

We took the naive approach:

- Google android emumator → Only x86_64
- Rooted to run Frida
- Headless to run on a server

Dynamic Analysis

We took the naive approach:

- Google android emumator → Only x86_64
- Rooted to run Frida → Easy to detect
- Headless to run on a server

Dynamic Analysis

We took the naive approach:

- Google android emumator → Only x86_64
- Rooted to run Frida → Easy to detect
- Headless to run on a server → Hard to debug

Dynamic Analysis

Additionnal issues:

- Random delays
- Random craches
- Sometime don’t start at all

SDK 34

The SDK 34 correspond to Android 14.

SDK 34

The SDK 34 correspond to Android 14.

Each new version of Android break things, like:

SDK 34

The SDK 34 correspond to Android 14.

Each new version of Android break things, like:

- You cannot load bytecode from a writable file

SDK 34

The SDK 34 correspond to Android 14.

Each new version of Android break things, like:

- You cannot load bytecode from a writable file
- You cannot write in a file whose name ends with `.dex`

Dataset

We used the dataset from a previous experiment (RASTA).

This dataset has too few malwares (only 5%).

Bytecode Loaded

Expectation: Malware dynamicaly load malicious code to spy on the user

Bytecode Loaded

Expectation: Malware dynamicaly load malicious code to spy on the user

Reality: Malware dynamicaly load advertisement librairies

Preliminary Results

Dataset

We tested the dynamic analysis on application from a subset of our RASTA dataset.

- 5 000 applications
- 5% of malwares
- Detected for the first time by Androzoo in 2023

Overview

	nb apk	nb failled	0 activity visited	at least one activity visited
All	4957	209	3860	888
With Reflection	3948		3152	796
With Code Loading	598		434	164

Why “0 activity visited”?

The application started, but did not display an activity (probably crashed).

Manual tests showed many possible reason:

- Bad application
- Google Play services not found
- Problem with Frida
- It works on my machine ™

Bytecode Loaded:

- 639 Bytecode file collected

Bytecode Loaded:

- 639 Bytecode file collected
- 273 identical files!

Bytecode Loaded:

- 639 Bytecode file collected
- 273 identical files!
- 492 files contain `Lcom/facebook/ads/*`
- 86 files contain `Lcom/google/android/ads/*`
- 48 files contain `Lcom/appsflyer/internal/*`

Bytecode Loaded:

- 639 Bytecode file collected
- 273 identical files!
- 492 files contain `Lcom/facebook/ads/*`
- 86 files contain `Lcom/google/android/ads/*`
- 48 files contain `Lcom/appsflyer/internal/*`
- Only 13 files, among 8 apks contains none of the above packages

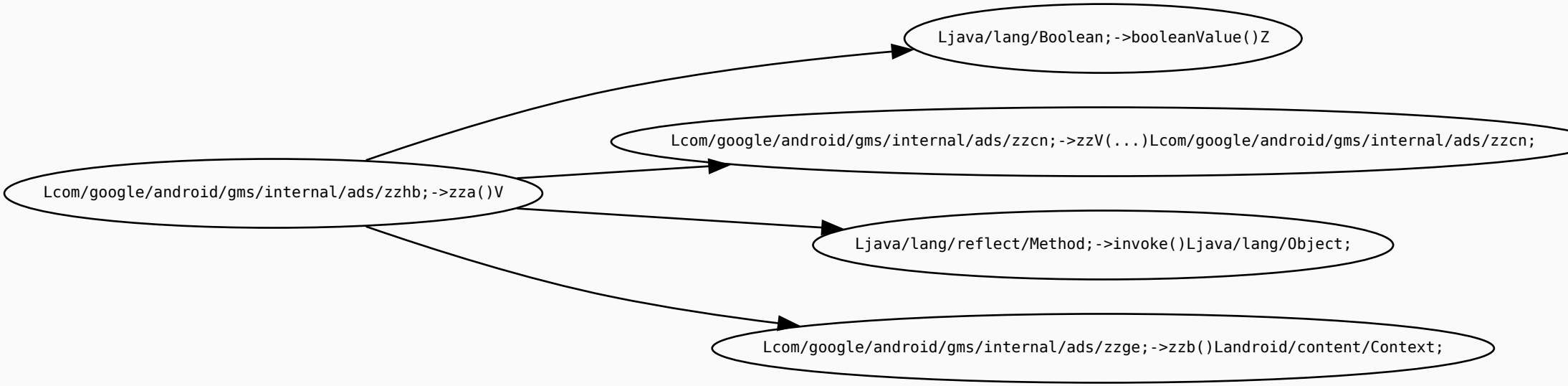
Real World Example: Call Graph

```
@Override // com.google.android.gms.internal.ads.zzhm
protected final void zza() throws IllegalAccessException, InvocationTargetException {
    try {
        this.zze.zzV(
          (
            (Boolean) this.zzf.invoke(null, this.zzb.zzb())
          ).booleanValue() ? zzdm.zzb : zzdm.zza
        );
    } catch (InvocationTargetException e) {
        this.zze.zzV(zzdm.zzc);
    }
}
```

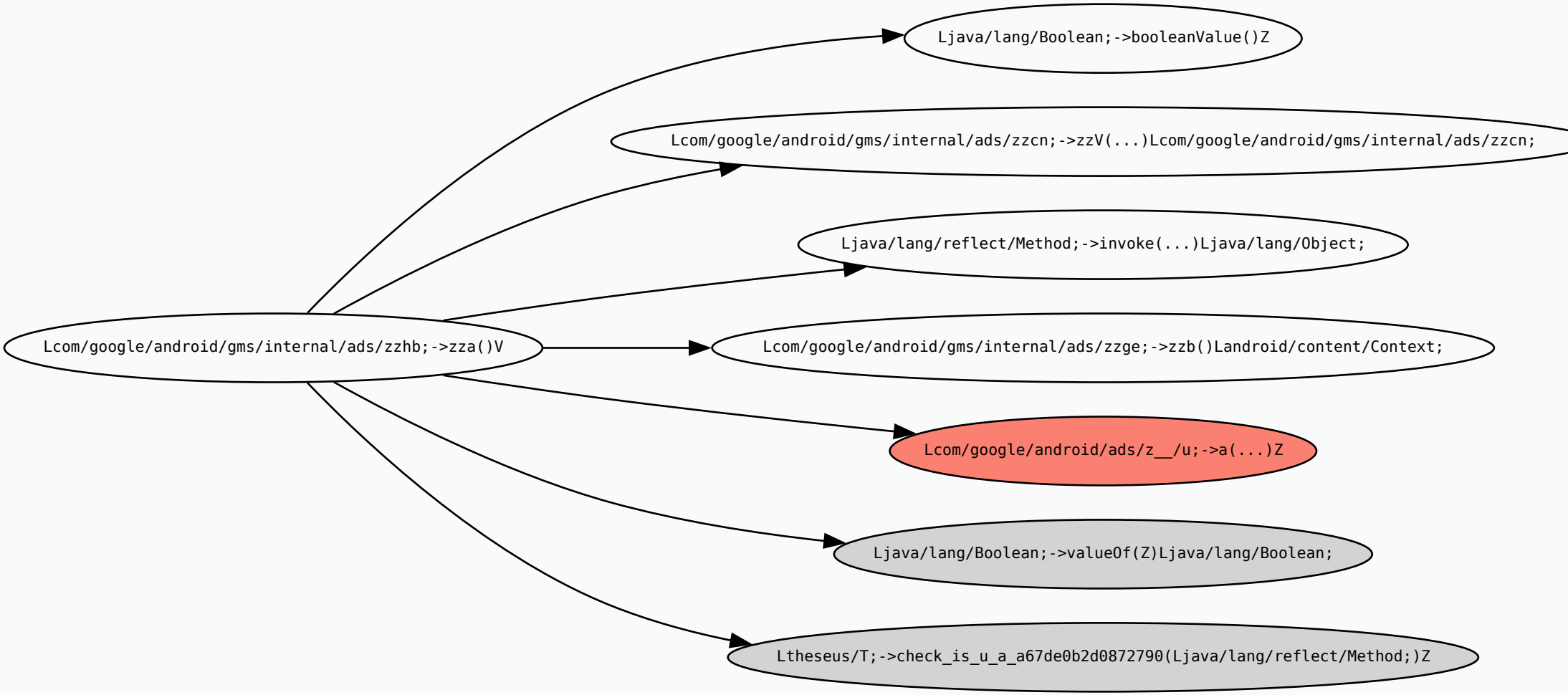
Real World Example: Call Graph

```
@Override // com.google.android.gms.internal.ads.zzhm
protected final void zza() throws IllegalAccessException, InvocationTargetException {
    try {
        Method method = this.zzf;
        Object[] objArr = {this.zzb.zzb()};
        this.zze.zzV(
          ((Boolean) (T.check_is_u_a_a67de0b2d0872790(method) ?
              Boolean.valueOf(u.a((Context) objArr[0])) :
              method.invoke(null, objArr)
          )).booleanValue() ? zzdm.zzb : zzdm.zza
        );
    } catch (InvocationTargetException e) {
        this.zze.zzV(zzdm.zzc);
    }
}
```

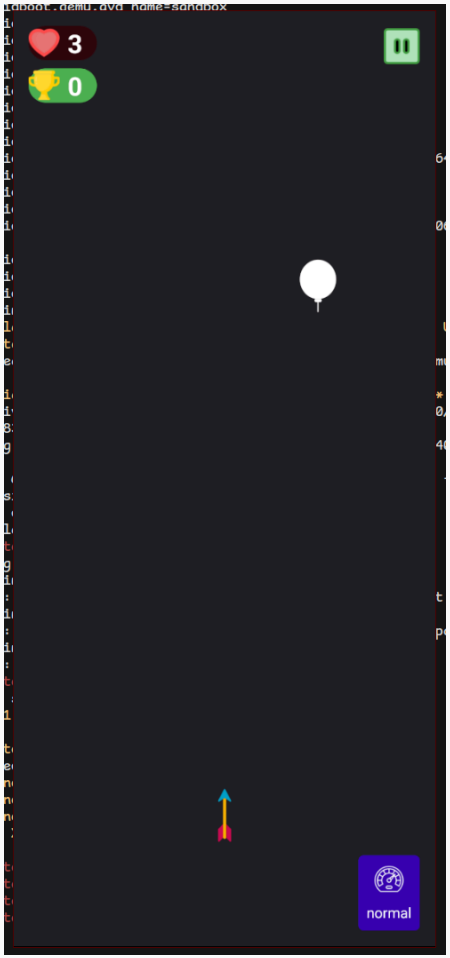
Real World Example: Call Graph



Real World Example: Call Graph



Real World Example: Emulator



Thanks You